2021 年 10 月 5 日
古関研究室 D2 三好正太
堀藤本研究室 2020 年度 M2 白戸柚衣

2021 年度電気電子情報実験第二　課題 19
制御系設計と運動制御　実験パートテキスト

目次

1　目的

2 日間のモータ制御実験を通して，システム同定から制御器設計，動作確認までの一連の制御器設計の流れを体験することが目的である。

1.1　実験１日目の目的

フィードバック制御の設計・実装ができるようになることが目的である。フィードバック制御器の設計法には Ziegler-Nichols の限界感度法のように，制御系が発散するまでゲインを上げて制御器を設計する経験則に基づく方法もあるが，発散させてはいけない制御対象も存在する。

本実験では，制御器設計のより進んだ方法として，システム同定により得た制御対象の情報を使って，制御器を設計したりシミュレーションする方法を学習する。このような制御系設計のステップはこのようになっている。今回の実験ではこのステップを一通り体験することを目標とする。

Step 1　制御対象物の物理的な理解
Step 2　システム同定
Step 3　制御器設計
Step 4　シミュレーション
Step 5　実験によるフィードバック制御の動作確認

1.2　実験２日目の目的

実験１日目で実装したフィードバック制御器に加えて外乱オブザーバを設計・実装する。例えば，モータを外から棒でつつく場合，この外からの力はモータを動かす力に対する外乱である。外乱オブザーバは外からの力に加えてモデル化誤差も外乱として扱い，まとめて抑圧する。外から棒でつつくとどのくらいの外乱が入ったのか記録できないので，外乱となる信号を作って加えることにする。外乱オブザーバにより外乱が抑圧される様子を観察する。

2　実験機の紹介

本実験で用いる実験機は，図 1 に示す 1 軸の Direct Drive(DD) モータ [3] である。このモータの実体は永久磁石同期電動機である。このモータの回転運動を制御することが本実験の目的である。

本実験で用いるモータは日本精工製型番 M-PS1018KN002 である。モータの電流コントローラとして日本精工製ドライブユニット型番 EDC-PS1018CB102-B を使用し，上位コントローラとしてオムロン製プログラマブル多軸コントローラ PowerPMAC CK3M[4] を使用する。PC 上でプログラムを書き換えて PowerPMAC に書き込むことにより，様々な運動制御が実現できる。

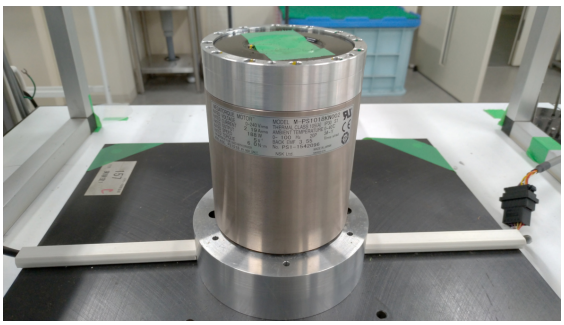


図 1　実験に用いる 1 軸 DD モータ

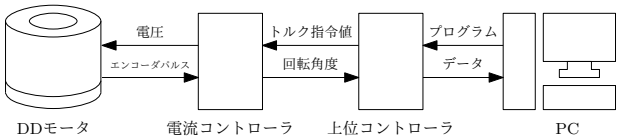


図 2　実験機器構成

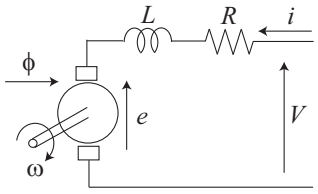


図 3　モータのモデル図

機器構成を図 2 に示す。

3　１日目の内容

3.1　制御対象物の物理的な理解

運動方程式を立てて制御対象物（プラント）がどういうものか把握する。これは Step 2 で実験したシステム同定が正しいか判断するのに有効である。また同定すべきプラントの伝達関数の次数などがわかる。

課題 1　他励直流モータ[*1]の電流から角速度までの伝達関数はどのように表されるか。電気的な回路方程式・モータの誘起電圧の式・機械的な運動方程式を考えて導出しなさい。図 3 はヒントとなる他励直流モータの図である。答えは図 4 を見れば簡単だが，なぜそうなるのか導出が大事である。

課題 1 で電流から角速度までの伝達関数を計算した

[*1] 本実験が対象とするモータは永久磁石同期電動機であるが，同期機や誘導機もインバータを用いたベクトル制御により直流機のように取り扱うことができる（このことは学部 4 年のモーションコントロールの講義で扱う）ので，ここでは直流機を制御対象とみなす。

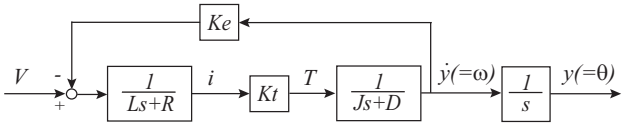


図 4　モータのブロック線図

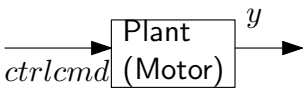


図 5　システム同定を行うためのブロック図

が，本実験では，電流制御を施し，モータの電流は電流指令値に瞬時に追従すると仮定する。この役割は 2 章の電流コントローラが担う。

このとき，電流指令値から電流までの伝達関数が 1 と見なせるので，電流指令値から角速度までの伝達関数は電流から角速度までの伝達関数と同じになる。

この実験では位置決めをしたいので，電流から位置までの伝達関数を考えると，

$$\frac{\theta(s)}{I(s)} = \frac{K_t}{Js^2 + Ds} \tag{1}$$

と表すことができる。

なお，本実験で使用するモータ，電流コントローラ系では，入力は電流指令値 I^* ではなくトルク指令電圧 T^* であり，トルク指令値電圧が 1 V の時トルク 0.98 Nm が出力されるように電流コントローラにより制御されている。電流コントローラが既に組み込まれているため，本 DD モータを制御する上では電流制御は隠蔽されており考慮することができない。従って，DD モータのトルク指令値入力から出力角度までの伝達関数は

$$\frac{\theta(s)}{T^*(s)} = \frac{1}{Js^2 + Ds} \tag{2}$$

と表される。

3.2　システム同定

Step 1 でプラントの伝達関数，本実験では式 (2) がわかっても，そのパラメータがすべてわかるとは限らない。そこで，システム同定によりパラメータを推定する必要がある。図 5 のようにプラント入力に電圧信号を入れて，プラントから出てくる信号を測定する。

3.2.1　信号の選択と信号の作成

システム同定の方法には，インパルス応答やステップ応答を入れて過渡応答を見る方法，サイン波を入れて周波数応答を見る方法などがある。過渡応答法はたたいたり，一定値を入れるだけなので簡単でよく使われるが，外乱・ノイズを分離できない，同定入力の振幅が精度を左右するなどの欠点もある。

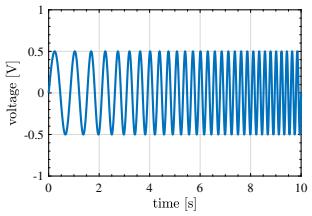


図 6　チャープサイン信号の例 (周波数 1-5 Hz, 振幅 0.5，周期 10 秒)

本実験では，周波数応答を見ることを考える。周波数応答を見る時，サイン波を入力する。サイン波では 1 つの周波数の情報しか得られないが，ボード線図を得るためには複数の周波数の信号が必要である。1 つの方法はある周波数のサイン波を一つずつ入力して，振幅と位相を取得する方法であるが，時間がかかる。別の方法はチャープサインといって，図 6 のように，実験時間の間に周波数を徐々に上げることで複数の周波数の信号を入力する方法である。周波数の低い領域も同定するためにはその分多くの時間をかけなければならないことに注意する。また，モータの定格に抵触するような大きな電圧を印加してはいけないことに注意する。本 DD モータではトルク指令値として ±9.8 Nm が最大定格である。普通はモータが壊れないように電流コントローラなどによりリミッタをかけているので，リミッタにより壊れる心配がなくても，波形がクリップされ正しく同定できなくなる。

課題 2　チャープサインは C 言語でどのように実装すればよいか。チャープサインの式を調べよ。

3.2.2　同定実験

課題 3　1Hz〜100Hz までのサイン波を，対数的に 7 点前後入力し，振幅と位相の情報を取得してボード線図を描け。

課題 4　課題 2 で作ったチャープサインをモータに入れてみよう。`ctrl_chirp.c` に C 言語を書き，`ctrl_chirp.h` で振幅・初期位相・はじめの周波数・終わりの周波数・1 周期の時間を指定する。メインのヘッダーファイル `rticplc.h` にチャープサインを印加する時間を書く。ノイズに強くするため，数周期測定する必要がある。本実験で用いる解析プログラムでは，入力として 4 周期のチャープサインデータが必要であることにも注意せよ。またチャープサインははじめの周波数と終わりの周波数の近くでは同定精度が落ちることに注意して周波数を決めること。さらに，チャープサインの分析を行う前に，制御入力が定格の範囲内かどうか，チャープサインの波形になっているか確認すること。

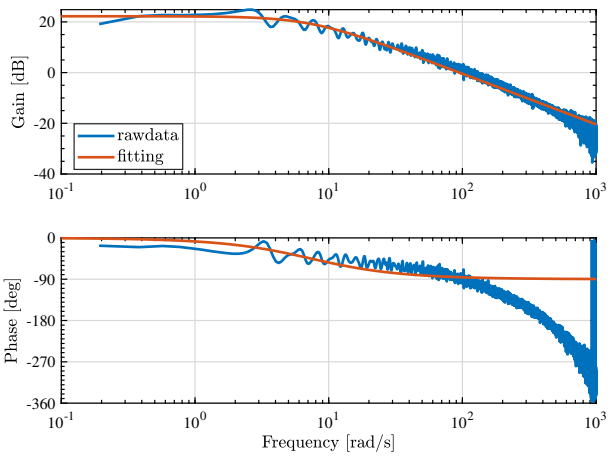


図 7　同定実験により得られたプラントのデータとフィッティング結果

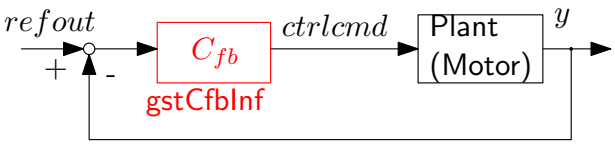


図 8　フィードバック制御を行う場合のブロック線図

課題 3 の各周波数点のデータと重ねて，システム同定結果が合うことを確認せよ。

3.2.3　同定実験結果の分析

測定した制御入力・出力の時間データからプラントの伝達関数を得るためには，時間データを周波数データに変換し，その後周波数データをフィッティングして，Step 1 で求めた伝達関数のパラメータを推定する必要がある。こうして得られたプラントの伝達関数を，以降，実際のプラントと区別して P_n と呼ぶ。

時間データを周波数データに変換したものとそのフィッティング結果は図 7 のようになるはずである。

課題 5　時間データを周波数データに変換するにはどのような処理を行えばよいか。日本語で説明せよ。

課題 6　フィッティングにより伝達関数のパラメータを推定しよう。周波数データをボード線図にプロットして，モータの伝達関数のボード線図と重ね合わせ，後者の係数を調整していく。自分でチャープサインの解析をするのは難しそうなので，`identification_chirp.m` を参照のこと。

3.3　制御器設計

プラントの伝達関数がわかったので，これを使って制御器を設計することが可能になった。図 8 の C_{fb} を設計することが目標である。

課題 7　フィードバック制御器を極配置法で設計してみよう。PD 制御と PID 制御を作ること。1, 2 日目課題で作った designPID, designPD 関数が役に立つであろう。

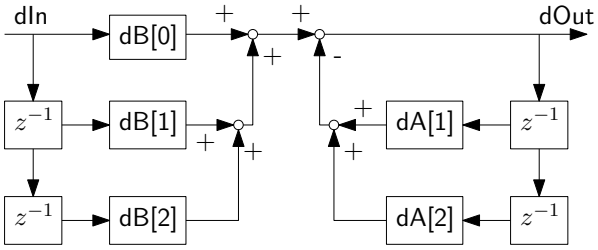


図 9　フィードバック制御器のブロック線図

3.4　シミュレーション

課題 8　3.3 節で設計したフィードバック制御系を Matlab/Simulink によりシミュレーションして結果をプロットせよ。ステップ入力を加えて出力がもし発散していたら閉ループ制御系の極を調整すること。また，制御入力が過大でないかも注意せよ。

3.5　制御器実装

3.5.1　制御器実装

制御器の設計やシミュレーションは連続系でできたが，実験機に実装する際には伝達関数の離散化が必要である。伝達関数の離散化には標準 z 変換

$$z = e^{sT}$$

や，Tustin 変換（双一次変換）

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

がよく用いられる [2]。それぞれの式において，T はサンプリング周期である。

3.3 節で設計した C_{fb} を離散化して係数を取り出し，C 言語の実験プログラム `ctrl_func.c` ファイルの構造体 `gstCfbInf` に反映させる。フィードバック制御器は図 9 のように実装する。図 9 の dB[0], dB[1], dB[2], dA[1], dA[2] は式 (3) の係数である。実はフィードバック制御器に限らず，分母が 2 次，分子が 2 次の形をしているものはすべて同じ形で実装できる。

$$C_{fb,z} = \frac{dB[0]z^2 + dB[1]z + dB[0]}{z^2 + dA[1]z + dA[2]} \tag{3}$$

課題 9　MATLAB を使って制御器を離散化し，制御器を C 言語で実装してみよう。本実験では Tustin 変換（双一次変換）で離散化することを考える。制御器の離散化は MATLAB で `c2d(sysc,Ts,'tustin')` を使うとできる。sysc は連続系の伝達関数である。制御周期 Ts は 1 ms である。

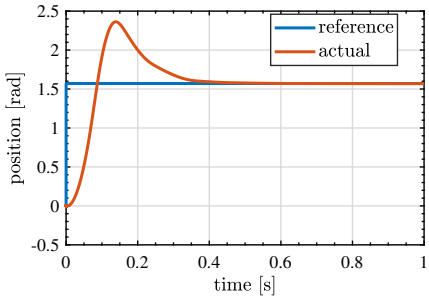


図 10　ステップ状の指令値に対する応答例 (青：指令値，赤：モータの位置)

3.5.2　フィードバック制御の実験

課題 10　図 10，11 は PD 制御または PID 制御の実験結果であり，モータの位置が指令値に追従している成功例である。自分が設計した制御器で位置がさまざまな指令値に追従できるか確認せよ。PD 制御と PID 制御で何か変わるだろうか。

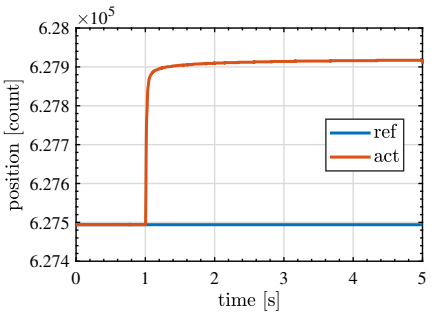
4　2 日目の内容

4.1　外乱オブザーバの設計

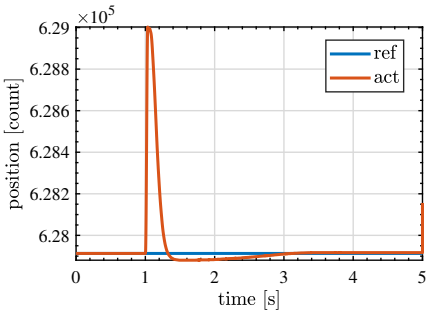
図 12 は外乱オブザーバのブロック線図である [5]。外乱オブザーバでは，モータに入力したつもりの制御入力 v_1 と測定できた出力 y の情報をトルクの次元で比較して，その差がトルク外乱であるとして，v_1 から引くことにより補償する。これにより，外からの力とモータのパラメータ変動による影響を外乱としてまとめて補償することができる。通常の積分制御よりも高速に応答し，外乱を強力に抑圧することが可能である。また，モータのパラメータ変動も外乱であるとして補償するため，モータの個体差に関わらず，与えたモータのノミナルモデル P_n の通りにモータの動きが振る舞うとして扱うことが出来るメリットがある。

外乱オブザーバの実験において加える外乱としては，例えばモータを棒でつつくこともそれに含まれるが，これでは毎回同じ外乱を加えることができないので，電圧 *vdistsim* を加える。システム同定でわかったように，プラントは分母の方が次数が高いので，出力の情報から電圧を復元するために逆数をとることはできない。そこで，分子の次数が分母の次数に比べて高くならないような次数のローパスフィルタ（LPF）を入れる。平等に比較するためにモータに入力したつもりの電圧からのパスにも LPF を入れる。

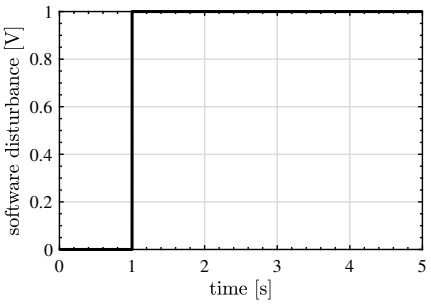
課題 11　Simulink で外乱を入れて外乱オブザーバがある場合とない場合を比較し，結果をプロットしなさい。



(a) PD 制御の実験結果（極配置の極は 20 Hz）



(b) PID 制御の実験結果（極配置の極は 10 Hz）



(c) ソフトウェア外乱

図 11　ソフトウェア外乱に対する応答 (青：指令値，赤：モータの位置)

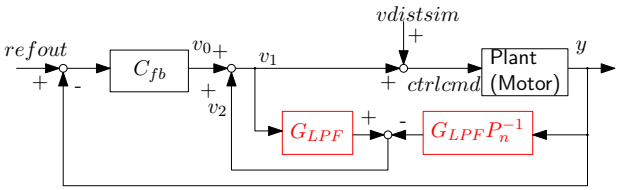


図 12　外乱オブザーバのブロック線図

4.2　外乱オブザーバの実装

シミュレーションしたものを実験で試してみよう。外乱オブザーバの実験をしたい場合は指令値を 0，外乱を入力するが，指令値はモータのもとの位置とするプログラムとすること。

課題 12　外乱オブザーバの G_{LPF} と $G_{LPF}P_n^{-1}$ を C 言語で実装できる形にしなさい。フィードバック制御器の実装と同じく，MATLAB で伝達関数を離散化して，係数を抜き出すこと。

課題 13　$v2$ を $v0$ に加えて外乱オブザーバとして一定の外乱を抑圧できるか実験しなさい。加えたソフトウェア外乱と外乱推定値をプロットし，正しく外乱推定できているか確認せよ。また，外乱推定値をフィードバックすることにより外乱が抑圧できることを確認せよ。外乱を抑圧できず発散する場合にはすぐに実験をやめ，理由を考察すること。

5　発展課題

発展課題 1　実験 1 日目には設計した制御器を用いてステップ応答で動作確認を行った。システム同定だけでなく設計した制御系の確認においても，周波数応答は時間応答と並んで重要である。設計した制御系の周波数応答からは，安定性と性能がわかる。例えば，開ループの伝達関数をプロットすれば (-1, 0) の点からどのくらい離れているかがわかる。また感度関数 $\frac{1}{1+CP}$ と相補感度関数 $\frac{CP}{1+CP}$ のボード線図の振幅から，低域で感度関数が低いかどうかやどの程度の周波数まで相補感度関数が 1 に近いか（帯域と言う）などがわかる。

図 9 の *refout* にチャープサインを入れて，入力 *refout* と出力 y から閉ループの周波数応答の解析を行いなさい。`rticplc.c` のフィードバック制御を行っている部分の指令値を step 指令から変更すればよい。チャープサインはシステム同定の実験を参考にして実装すること。

このとき，飽和していないか確認するために制御入力 *ctrlcmd* も記録すること。開ループ伝達関数や感度関数，相補感度関数をプロットして，そのようになっているとなぜ良いのかも考察すること。

発展課題 2　ステップ指令値の後に LPF を挿入すると，ステップ応答をなめらかにし，オーバーシュートもなくすことができる。設計したフィードバック制御系において，オーバーシュートをなくすためにはどのような LPF を設計すればよいだろうか。オーバーシュートが発生する原因を考えて，極と零点の位置から考察しなさい。できれば実験してみること。実験する場合は構造体 gstLPFInf の係数を変えること。システム同定によりプラントは 2 次で得られているので，設計したフィードバック制御器を使ってシミュレーションを行ってもよい。コードは `student_kadai_advanced` フォルダの `kadai_advanced_lpf.m` にある。LPF のカットオフ周波数を 200 Hz から変えること。

発展課題 3　フィードバック制御のみでは，内部モデル原理により達成できる目標値追従性能に限界があることが知られている。特に目標値追従性能と外乱抑圧性能の両立が問題となる。フィードフォワード制御を用いる 2 自由度制御とすると，外乱抑圧性能を損なわずに目標値追従性能を改善できる。実験プログラムにフィードフォワード制御器を実装し実験してみよ。

発展課題 4　モータのモデル化で電流制御を行えば電流は電流指令値に追従できると述べた。電流制御とはどのようなものか，ブロック線図を用いて説明せよ。

発展課題 5　この他，自由な発想で実験を考えて行ってみよ。

6　実験機を動かす手順

実験機を動かす手順を説明する。
実験機を動かすために必要な手順は大別すると

1. 実験機の起動
2. 実験プログラムの編集
3. 実験の実行

からなる。順を追って説明する。図 13，14 も合わせて参照すると良い。

6.1　実験機の起動

実験機を起動する手順を示す。

1. PC の電源を入れる。ID とパスワードは PC のある机に貼ってある付箋を見てください。
2. モータドライバから出ている SVON スイッチを OFF にしておく。
3. 実験機（PowerPMAC と電流コントローラ）の電源を入れる。電流コントローラについている 7 セグメント LED の表示が「o」であることを確認する。それ以外の表示の場合は電流コントローラがエラーを出力している。TA に報告し指示を仰ぐ。
4. PowerPMAC IDE を起動し，PMAC と PC が接続されることを確認する。

6.2　実験プログラムの編集

実験プログラムの雛形をダウンロードし，編集する手順を示す。図 13，14 も併せて参照すると良い。

1. 実験プログラムを `https://github.com/eeicpower/EEIC_Bachelor_Motion_for_Student_2020` からダウンロードする。
2. PowerPMAC IDE を起動し，ソリューション `EEIC_Bachelor_Motion_for_students` を開く。ソリューションのディレクトリは `EEIC_Bachelor_Motion_for_Student_2020/EEIC_Bachelor_`

Motion_for_students/EEIC_Bachelor_Motion_for_students.PowerPmacSuite_sln である。

3. 編集すべきファイルは C Language フォルダ中の
   - Libraries/ctrl/ctrl_chirp.c
   - Libraries/ctrl/ctrl_func.c

   の 2 つである。ctrl_chirp.c においてチャープサインの実装を行い，ctrl_func.c において種々の制御器の実装を行う。

6.3 実験の実行

実験プログラムの編集を終わり，実験を実行する際の手順を説明する。

1. PowerPMAC IDE 内のターミナルを起動し，disable rticplc と入力する。これは実験プログラムを動かさないためのコマンドであり，意図しないプログラムの起動を防ぐ。
2. ターミナルに P50 = 0 と入力する。これは本実験プログラムの動作において何もしないを選択する操作である。
3. ソリューションエクスプローラでソリューション名を右クリック→「すべてのプログラムをビルドしてダウンロード」を選択する。ビルドに成功すると PMAC にプログラムがダウンロードされる。次の手順へ進む。ビルドに失敗した場合はエラーメッセージを読んでプログラムのデバッグをする。実験プログラムの編集に戻る。
4. PMAC にプログラムがダウンロードされたら，SVON スイッチを ON にする。
5. ターミナルに enable rticplc と入力する。これは実験プログラムを動かすためのコマンドである。
6. 「プロット」ウインドウを開く。
7. 「プロット」ウインドウで取得したいデータ，P1 P2 P3 P5 P6 P7 P9 P10 P98 P100 の 10 個の P 変数をリストに入れる。それぞれの P 変数の意味は rticplc.c において P 変数が定義されている箇所を参照されたい。プロットの最大収集サンプルを一番右側に，サンプリング設定のサンプル期間を 1 にする。以上の作業が終わったら「データの収集」ボタンを押す。
8. P50 = 1,2,3 を入力する。P50 変数に設定する値により実行する実験を選択する。詳しい動作は 8 章を参照せよ。
9. 実験が終わったら「プロット」ウインドウを開き，「データのアップロード」を押す。ここで間違って「データの収集」を押すと実験データが消えるので注意せよ。
10. 保存したいデータをすべて同じグラフにプロットして，そのグラフの画面から「txt データとして保存」を選択する。[*2]
11. 実験が終わったらターミナルに P50 = 0 と disable rticplc を入力する。
12. 続けて実験したいときはターミナルに disable rticplc，enable rticplc をこの順で入力する。PowerPMAC の時刻データがリセットされ，再実験できるようになる。
13. PowerPMAC がエラーを出力したときなどにリセットをする場合はターミナルに$$$を打ち込む（エラーが出たら TA を呼ぶこと）。

6.4 安全上の注意（必ず読む！）

実験機は制御器が発散すると暴走することがあるので，すぐに非常停止スイッチを投入できるように準備してから実験を開始する。

PID 制御や外乱オブザーバの実験において棒で DD モータをつつく場合，棒の後ろに人間が来ないようにする，棒の持ち方に注意する。

感染症対策のため窓を開けて換気をする。

7 実験データ解析の手順

実験により得られたデータを解析する手順を説明する。

実験データの解析は

1. txt ファイルの mat ファイルへの変換
2. mat ファイルに対する MATLAB によるデータ処理

からなる。

7.1 txt ファイルの mat ファイルへの変換

student_expdata_plot/namedef.m を用いる。スクリプトを実行すると入力ファイルを選択するダイアログが出てくるので，mat に変換したい txt ファイルを選択する。処理が完了すると，matfiles フォルダに txt と同じファイル名の mat ファイルが作成される。

7.2 mat ファイルに対する MATLAB によるデータ処理

システム同定には student_expdata_plot/identification_chirp.m を用いる。システム同定に用いるチャープサインの時間や周波数範囲を変更した際は，m ファイルの該当するパラメータを変える必要

[*2] 白戸さんいわく，EEIC_Bachelor_Motion_for_students/Documentation フォルダに保存することを推奨する。他のディレクトリに保存すると保存に失敗することがあるらしいが，三好が実験したところ再現しなかった。

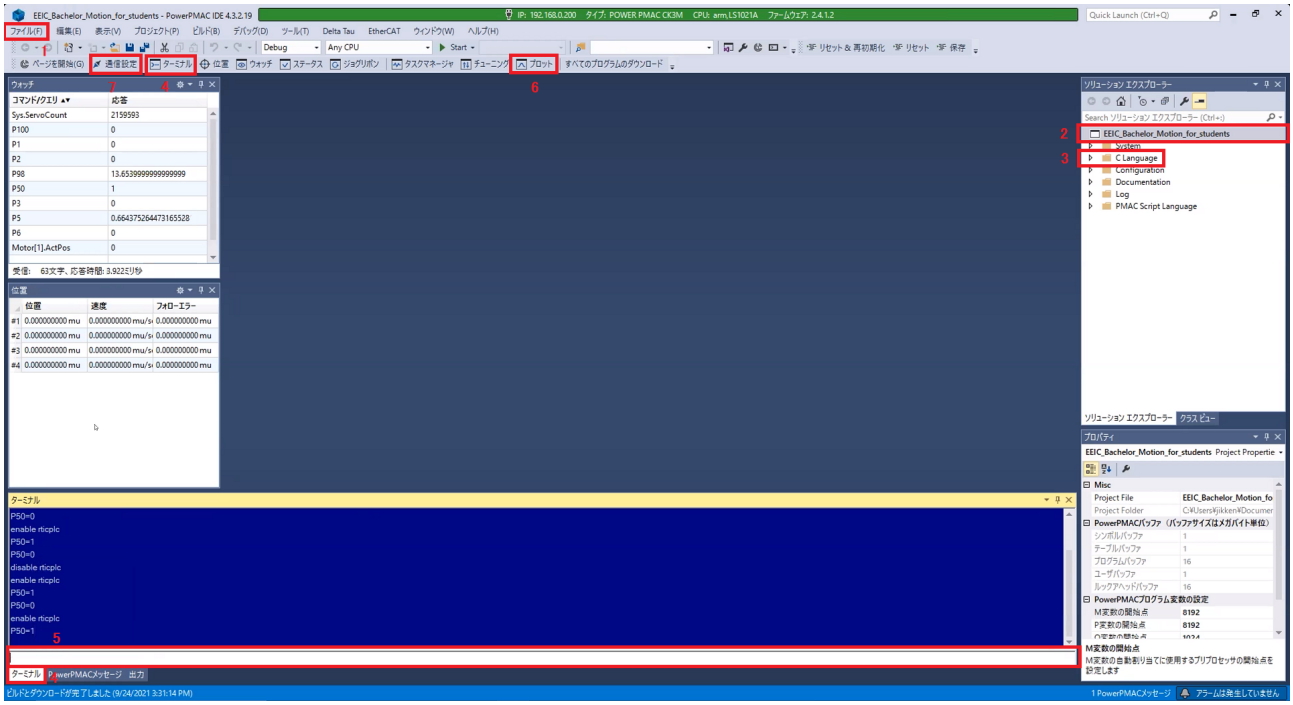


図 13　PowerPMAC IDE の画面と，使用する機能の説明。1. ファイルメニューから PowerPMAC プロジェクトを開く。2. ソリューション名を右クリック→すべてビルドしてダウンロードによりプログラムのコンパイルと PMAC への転送を行う。3. C Language 以下に編集すべき C ソースファイルがある。4. 「ターミナル」ボタンを押すことで画面下方にターミナルを出現させる。5. ターミナルにコマンドを打ち込む。6. データの保存に用いる「プロット」ウインドウを表示する。7. PowerPMAC との通信トラブルが発生したときは通信設定を確認する。

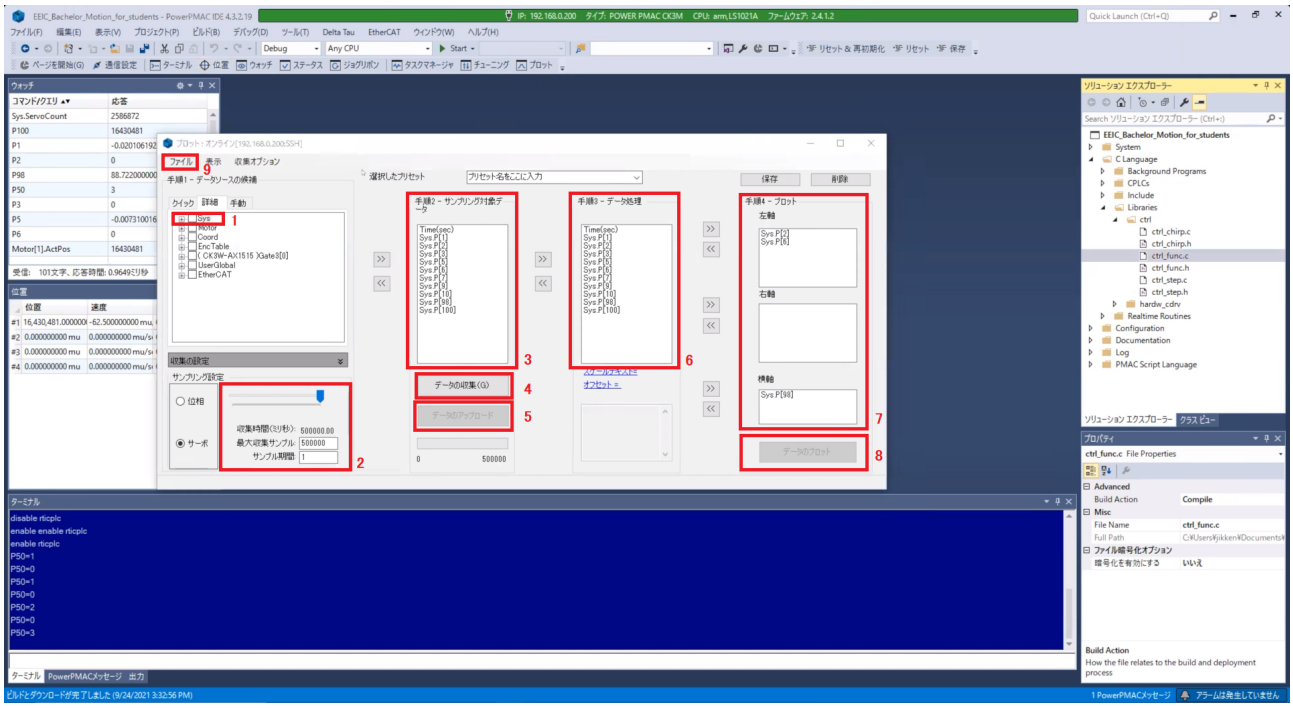


図 14　PowerPMAC IDE のプロットウインドウの説明。1. 取得する変数を選択する。Sys.P 変数は Sys の中にある。2. 最大収集サンプルは 500000，サンプル期間は 1 にしておく。3. サンプリング対象データに含まれる変数の値が取得される。4. 実験を実行する直前にデータの収集を開始する。もう一度押すとデータの取得を終了する。5. データの取得終了後データのアップロードを押すとデータのプロット，（上書き）保存ができる。6. データ処理に含まれるデータが保存される。サンプリング対象データと同じにしておく。7. 保存あるいはプロットするデータを選択する。8. データをプロットする。データを保存する際には保存したいデータを一度プロットする。9. プロットが表示される画面において，ファイル→生データの保存でデータを txt ファイルとして保存する。

がある。

PD 制御，PID 制御，外乱オブザーバの実験結果のプロットは plot コマンドなどを使用して自分で頑張って書く。

8 実験プログラムの説明

PMAC プロジェクトの紹介をする。例えばチャープサインの周波数などは ctrl_chirp.h，フィードバック制御器のパラメータは ctrl_func.c の中で変更する。変更する可能性があるファイルは次のファイルである。rticplc.c 以外は Libraries フォルダに入っている。

- rticplc.c　メインの処理が書かれた C 言語ファイル。
- ctrl_chirp.c チャープサインを実装するファイル。
- ctrl_func.c　フィードバック制御器の関数が入ったファイル。

実験のためにシステム同定，普通の制御，外乱オブザーバを用いた制御が用意されている。それぞれ rticplc.c 中の flag_exptype = 1，2，3 の部分である。rticplc.c の上の方で処理をしたので，ターミナルで P50 = 1 と打ち込むと flag_exptype = 1 の部分のコードが実行される。

8.1 P50 = 1, チャープサインを用いたシステム同定

システム同定はプラントに直接電圧を入力するので，ctrl_chirp.c でチャープ状の電圧を作っている。チャープサインの設定を変えるには ctrl_chirp.c, ctrl_chirp.h を変更する。その後時間の処理をしている。サンプリング周期を 1 KHz にしているので，プログラムが 1 回回るごとに時間を 0.001 秒進ませている。電圧を出力する処理は他の実験にも共通するので rticplc.c の下部にある。

8.2 P50 = 2, PD 制御・PID 制御

90 deg 回転するようなステップ指令を与えている。ステップ指令は出力と次元が合うように変換された後，指令値と出力の差が制御器に入力される。PID 制御は func_TF2Exe で，PD 制御は func_TF1Exe で行われる。制御器は構造体 gstCfbInf[0] である。この制御器は rticplc.c の上部で初期化されている。この制御器の係数を変えるには ctrl_func.c を変更する。制御器の出力は電圧で rticplc.c の下部で出力される。

8.3 P50 = 3, 外乱オブザーバを入れた制御

P50 = 2 の場合と同じようにフィードバック制御のプログラムであるが，外乱オブザーバの部分が異なる。係数は ctrl_func.c で変更する。

9 MATLAB ファイルの説明

配布する MATLAB コードは以下のフォルダに入っている。

- student_expdata_plot フォルダ
  1 日目 Step2（システム同定），Step5（フィードバック制御器の実験），2 日目（外乱推定・補償）に使用。
- student_ctrldesign フォルダ
  1 日目 Step3（制御器設計），2 日目外乱オブザーバ設計に使用。
- student_simulation フォルダ
  1 日目 Step4（シミュレーション）に使用。
- student_kadai_advanced フォルダ
  発展課題 2 に使用。

9.1 student_expdata_plot フォルダ

- namedef.m
  テキストファイルで出力されたデータを mat ファイルとして保存するためのファイル。スクリプトを実行すると入力ファイルを選択するダイアログが出てくるので，mat に変換したい txt ファイルを選択する。処理が完了すると，matfiles フォルダに txt と同じファイル名の mat ファイルが作成される。測定データの単位系の [count]→[rad]，[V]→[Nm] の変換もこのスクリプトで行われる。
- identification_chirp.m
  システム同定するためのファイル。システム同定に使うデータは上の namedef.m で mat ファイルにしていたものを使う。システム同定の周期，周波数などを変えた場合は該当箇所を変えること。
- plot_fbresult
  フィードバック制御系のプロットをするためのファイル。データは上の namedef.m で mat ファイルにしていたものを使う。

9.2 student_ctrldesign フォルダ

student_expdata_plot フォルダで同定したシステム同定結果を使う。システム同定では速度までを同定したが，位置までがプラントなので，積分をかけるのを忘れないように気をつける。システム同定がうまく行かなかった場合はデフォルトで入っている値（以前 TA が別のモータを用いて図 7 のように同定した結果から遅れを無視したもの）を使う。C 言語の ctrl_func.c において Cpid, Cpd, LFmath, INVQmath の係数を指定してい

るところがあるので，そこを変更する。dB は分子，dA は分母であり，上から分子最高次の係数，分子で二番目に次数が高い項の係数の順で埋めていく。例えば，2 次の伝達関数であれば

$$f(z) = \frac{dB[0]z^2 + dB[1]z + dB[2]}{dA[0]z^2 + dA[1]z + dA[2]} \tag{4}$$

の構造である。

- `ctrldesign_PID.m`
  PD，PID の設計に使用する。
- `ctrldesign_dist.m`
  外乱オブザーバの設計に使用。LFmath は入力側からの LPF の伝達関数 G_{LPF}，INVQmath は出力側からの LPF の伝達関数とプラント逆モデルの積 $G_{LPF}G_{\mathrm{nominal}}^{-1}$ である。

9.3 `student_simulation` フォルダ

`student_ctrldesign` フォルダで設計した制御器を使う。

- `sim_pid`
  PID, PD 制御器のシミュレーション。
- `sim_dist`
  外乱オブザーバのシミュレーション。

参考文献

[1] 足立，MATLAB によるシステム同定，東京電機大学出版局，1996

[2] 原島，堀，工学基礎 ラプラス変換と z 変換，数理工学社，2004

[3] メガトルクモータ | 製品情報 | 日本精工 (NSK) https://www.nsk.com/jp/products/megatorque/（2020 年 11 月 3 日閲覧）

[4] プログラマブル多軸モーションコントローラ CK3M / CK3W - 商品カテゴリ | オムロン制御機器 https://www.fa.omron.co.jp/products/category/automation-systems/multi-axis-controller/programmable-multi-axis-controller-ck3m_ck3w/（2020 年 11 月 5 日閲覧）

[5] 大西 公平, 外乱オブザーバによるロバスト・モーションコントロール, 日本ロボット学会誌, 1993, 11 巻, 4 号, p. 486-493, https://doi.org/10.7210/jrsj.11.486